SOFTWARE MAINTENANCE: RESEARCH AND PRACTICE, VOL. 9, 345–364 (1997)

Research

Using Risk Analysis to Manage Software Maintenance

SUSAN A. SHERER*
Lehigh University, College of Business and Economics, 621 Taylor Street, Bethlehem PA 18015, U.S.A.

SUMMARY

Software maintenance management can be improved through explicit consideration of the impact of maintenance efforts on project risk, maintainability risk and usability risks, including functionality, performance, financial and software failure risk. The paper illustrates how to analyse the effect of maintenance on software failure risk. I also show how risk analysis can guide important maintenance decisions: determining whether to redevelop software, prioritizing maintenance efforts, allocating resources and planning release schedules. © 1997 John Wiley & Sons, Ltd.

J. Softw. Maint.: Res. Pract., **9**, 345–364 (1997)

No. of Figures: 3. No. of Tables: 7. No. of References: 39.

KEY WORDS: software failure risk; project risk; maintainability risk; usability risk; maintenance management; analysis of risk

1. INTRODUCTION

The management of maintenance continues to challenge the software industry (Henry, Blasewitz and Kettinger, 1996). Although it consumes 60–80% of information system (IS) resources (Bassett, 1995; Vogel, 1996), software maintenance remains 'the Cinderella' of the IS department (Vowler, 1992). Effective management techniques are needed to manage risks associated with maintenance activities.

Maintenance is a fundamental part of the life cycle of software. The ability of software to change and evolve distinguishes it from hardware, making it attractive to meet the demands of rapidly changing environments. While some maintenance includes *correction* or *prevention* of errors in design, specifications and coding, other activities encompass *adaptation*, changes to accommodate alterations to data, files, hardware, software and policies, and *perfection* through enhancements (Swanson, 1976). Since software cannot be guaranteed to be 100% reliable, corrective and preventive maintenance continue to be required throughout the software life cycle. Both adaptive and perfective maintenance enable software to evolve with changes in the environment. Changeability is an inherent

* Correspondence to: Susan A. Sherer, Lehigh University, College of Business and Economics, 621 Taylor Street, Bethlehem PA 18015, U.S.A. E-mail:sas6@lehigh.edu

CCC 1040–550X/97/060345–20$17.50

Received 8 April 1997
Revised 3 September 1997

property of modern software systems (Brooks, 1987). If software could not be changed, we would need to redevelop systems every time the environment changed, an infeasible practice given the high costs of development (Lanubile and Visaggio, 1995).

But, maintenance itself is very costly. The systems base in most organizations is increasing, escalating maintenance costs. Some organizations are so overwhelmed with maintenance that they no longer have the time or resources to develop new applications (Moad, 1990). Some systems are too outdated to be maintained; redevelopment becomes a more attractive alternative. At the same time, maintenance introduces new risks to an IS organization. While maintenance projects have similar risks to development projects, they are exacerbated by the need to incorporate changes into existing systems that are often complex and difficult to comprehend. Effective management techniques are needed to control costs and manage risks.

The objective of this paper is to demonstrate how managers can measure the effect of proposed maintenance changes on risk and use this information to guide software maintenance management decisions. Section 2 describes the different types of risk that must be effectively managed by software maintenance managers. Section 3 then focuses on one important component of risk, describing methodology to measure software failure risk, the expected loss due to software failure. Section 4 introduces procedures that managers can use to measure the effect of maintenance on software failure risk. Section 5 demonstrates how information about the effect of maintenance on different types of risk can be used to plan, organize and control maintenance activities. We show how risk impact analysis can guide important maintenance decisions: determining whether to fix or redevelop software, prioritizing maintenance efforts, allocating resources and planning release schedules.

2. MAINTENANCE RISKS

The maintenance process introduces many risks. These risks are similar in type to those faced by new software developers, but degree of risk varies. Because maintainers must interact with existing systems that are often complex and difficult to comprehend, several risks are increased.

The three types of risk in software maintenance are:

- *Project risk*—maintenance project cannot be completed because personnel lack capabilities, software cannot be maintained on time or within budget, organization does not have an effective maintenance process.
- *Usability risk*—maintained systems will cause problems when used.
- *Maintainability risk*—system will be difficult to maintain in the future once changes are made.

One component of project risk is the lack of personnel to adequately carry out maintenance tasks. Several personnel problems that particularly plague maintenance efforts include low morale due to lack of recognition and respect afforded maintenance, high turnover causing loss of expertise, lack of experienced personnel, minimal training, and inadequate understanding and response to business needs (Deklava, 1992). Alternative

organizational structures and aggressive training have been suggested to manage these risks (Pigoski and Looney, 1993; Swanson and Beath, 1990).

Process models have been developed to manage the risk when the organization does not have an effective process for maintenance (Henry, Blasewitz and Kettinger, 1996; Hinley and Bennett, 1992, 1993). The Capability Maturity Model can assess an organization's software maintenance processes (Drew, 1992) although, in practice, it is not widely applied (Taylor and Wood–Harper, 1996). Minimizing cost/schedule risk involves accurate estimation models (Sneed, 1995) and project management tools including cost/schedule metrics (Stark, 1996).

Usability risk includes:

- Functionality risk—system does not provide needed functionality;
- Software failure risk—system fails due to software faults;
- Performance risk—system does not meet real–time performance standards; and
- Financial risk—system does not achieve financial benefits.

Usability risk often plagues a maintained system that suffers obsolescence from decline in quality or loss of revenue from not using improved substitutes (Sakthivel, 1994).

Maintainability is a function of modularity, descriptiveness, consistency, simplicity, expandability and instrumentation (Peercy, 1981). When we maintain software, it grows in size and complexity with a decrease in understandability and adaptability (Pfleeger and Bohner, 1990). Maintainability risk results from deterioration of software, degradation from piecemeal maintenance changes and obsolescence resulting from not using the latest technological developments to reduce future software maintenance costs (Sakthivel, 1994). Deterioration affects not only the code but the documentation as well (Turver and Munro, 1994). Maintainability metrics can gauge the effect of maintenance on systems (Ash *et al.*, 1994; Pearse and Oman, 1995).

While there is some overlap among these types of risk, they are primarily distinguished by the time period during which loss affects the organization and the risk agents. Project risk impacts the organization when the maintenance project is taking place, usability risk affects the organization when the project is complete and the system is used, while maintainability risk affects the organization when it attempts to further modify the system. Maintainability risk is primarily the producer's (or maintainer's) risk while usability risk is primarily the user's risk (Leung, 1996). Project risk is shared by both developer (producer) and user.

Software maintenance is aided by change impact analysis, the activity of identifying not only what to modify to accomplish a change, but also identifying the potential consequences of a change (Arnold and Bohner, 1993). Traceability and dependency analysis find relationships among all software life cycle objects and program entities, respectively (Bohner and Arnold, 1996, p. 2). By identifying what needs to be changed, the complexity of the maintenance task is assessed. This is especially useful in assessing project risk, providing estimates of change effects on resources, effort and schedule (Pfleeger, 1991, p. 433), in particular identifying high risk ripple propagators (Turver and Munro, 1994). However, change impact analysis does not usually measure the consequence or the impact of proposed changes on usability risk. We extend impact analysis by

providing specific methodology to assess the impact, or effect, of maintenance on software failure risk. *Software failure risk impact* is the estimated amount of change in software failure risk, the expected value of loss resulting from software failure, as a result of maintenance efforts. It measures both changes in failure likelihood and magnitude of loss resulting from failure.

3. SOFTWARE FAILURE RISK

We summarize here a methodology for assessing software failure risk, the expected loss resulting from software failure (Sherer, 1992a, pp. 55–85). The methodology involves four stages leading to the assessment of module failure risk, as shown in Table 1.

External risk assessment involves a focused study of the environment in which software will operate to anticipate scenarios that can result in hazards or potential situations that can lead to economic loss. The risk assessor must understand how users will interact with the software and what an organization will do with the software's outputs. Hazard scenarios are developed that consider all external events leading to loss, potential actions (or inaction) that can contribute to loss, and all relationships among environmental conditions, hardware failures, human interactions and erroneous information and control procedures. Both fault and event trees can help depict hazard scenarios (Sherer, 1992a, pp. 62–69). A hazard's consequence is assessed by weighting each scenario's potential loss by its likelihood, a function of the environmental control risk, the degree to which user actions and environmental conditions fail to prevent loss.

Module exposure is the expected magnitude of loss due to faults in a particular module. A module is a component or related subset of program instructions. Module exposure depends upon the module's relationship to external risks, a function of the use and processing of that module. Loss may result if a module's function and use are related to a scenario resulting in a specific hazard—i.e., the hazard is included in the module fault potential set. Expected module loss is estimated by summing the probability of each module use, weighted by the expected consequence of all hazards that may result from

Table 1. Module failure risk assessment

Stage	Description	Steps
External risk assessment	Analysis of the environment external to the software	1. Identify hazards 2. Hazard scenario analysis 3. Consequence analysis
Module exposure assessment	Study of software to determine expected magnitude of loss due to failures caused by faults in each module	1. Module fault potential 2. Module use distribution 3. Hazard probability distribution
Failure likelihood estimation	Estimate of expected number of failures from faults in modules	1. Estimate number faults, per fault failure rate 2. Software reliability model
Module failure risk assessment	Expected loss resulting from software failure caused by faults in a module	Module exposure times expected number of failures

that use. Expected consequence of all hazards for a given use weights the consequence of each hazard in the module fault potential set by the hazard probability distribution, the probability that a given hazard occurs for that use of the module. The hazard probability distribution can be assumed to be uniform for all faults in the module fault potential set or weighted by the severity of the loss.

The *failure likelihood*, the expected number of failures during time t due to faults in a module, is estimated with a time–dependent software reliability model. The model parameters, mean number of faults and per fault failure rate, are estimated from module size and complexity. We use the finite failures, Poisson, exponential software reliability model (Musa, Iannino and Okumoto, 1987, p. 285):

$$N(t) = \mu\,(\,1 - e^{-\phi T}) \tag{1}$$

where μ = number of faults in a module, ϕ = per fault failure rate and T = execution time of module during operational time t.

Module risk is estimated as the product of module exposure and expected number of failures. This estimate may be exaggerated because each module is assessed for the loss for all the hazards to which its potential faults may relate even when other modules could detect and correct for these errors. Note that this estimate of risk can include both producer risk as well as user risk (Leung, 1996) if the exposure estimates include the cost of fixing the software.

Software failure risk assessment can guide the development of new software. Maintenance activities can either increase or decrease module failure risk. Evaluation of the impact of these activities on software failure risk can guide maintenance management decisions.

4. FAILURE RISK ASSESSMENT IN MAINTENANCE

4.1. Module failure risk

Now that we have described the methodology for assessing software failure risk, we focus on the assessment of the impact of maintenance on software failure risk. We suggest that proposed maintenance efforts be analysed to understand their impact on external hazards, module exposure and failure likelihood.

Module failure risk should be estimated for each module in a system as it is being developed. These estimates are updated as software is used (Sherer, 1992a, p. 175). When a specific maintenance request is received, the impact of this maintenance on failure risk can then be evaluated. Software failure risk impact is the effect of maintenance efforts on the probability of failure of the software and the magnitude of loss resulting from software failure.

4.2. External failure impact analysis

We first consider the effect of proposed maintenance on the external use of a system. Enhancements may introduce new external hazards. Existing hazards may be removed or

revised if the expected results of maintenance efforts change the way the system will be used. Figure 1 describes steps in evaluating the impact of maintenance on external failure risk.

External failure impact analysis should be accomplished during the first phase of maintenance, the analysis of software change impact, during which time current system work products, the change request and previously known ripple effects are analysed to

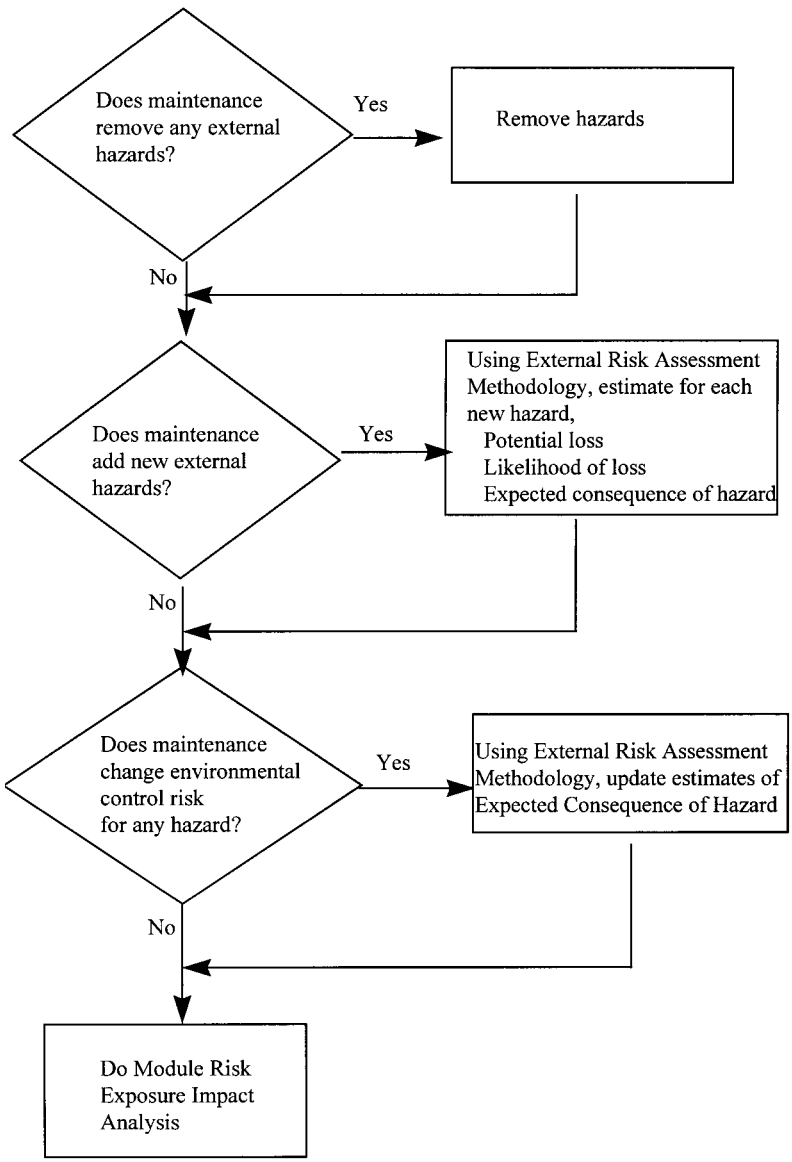


Figure 1. Procedure for external failure impact analysis

determine traceability of change requirements. Traditional impact analysis supports release planning by identifying software life cycle objects that are likely to change for each software change proposed in a set of change requests. The analysis is often used to develop precise effort and cost estimates of software changes (Bohner, 1996) to manage project risk. External failure impact analysis provides additional information that can be used to manage project risk. Performing this analysis early in the maintenance process can provide information that could alter effort and cost estimates. If maintenance requirements introduce hazards with high consequence, then project schedule might be lengthened to allow more time for testing changes. External failure impact analysis may also suggest alternative approaches to implementation of maintenance requests. These approaches may alter control risks and should be considered when hazard consequence is very high.

4.3. Module exposure impact analysis

Once software change impact is analysed, we typically specify and design the software change (Bohner, 1996). If potential degradation of the system is unacceptable, the necessity for change may be reassessed, or the proposed implementation of the change may be re-evaluated (Bohner, 1996). Module risk assessment can aid this analysis by evaluating not only the impact on failure likelihood, but changes in exposure as well.

We use the same module exposure assessment procedure for new modules to be added that we use for new systems development. To update exposure assessment for modules that will be revised during maintenance, we consider whether a module relates to any new hazards, or hazards with adjusted consequences, and whether the module has new uses. We can use both traceability and dependency analysis to identify modules related to new or revised hazards. Figure 2 shows the procedure for assessing the impact of maintenance on module exposure for revised modules.

4.4. Failure likelihood impact analysis

To assess failure likelihood for new modules, we use the same assessment methodology that we use for new systems (Sherer, 1992a, pp. 55–85). Reliability estimates for updating existing modules are adjusted using procedures outlined by Musa *et al.* (1987, pp. 440–444). These procedures assume that faults are introduced only in newly written or modified code and that code is not removed. The expected number of failures in a given time period is then estimated with the Poisson exponential time translated software reliability model (Musa, Iannino and Okumoto, 1987, pp. 436–440).

If failure likelihood is high, particularly for high exposure modules, software redevelopment may be a viable maintenance strategy. Software redevelopment can reduce failure likelihood by providing software with fewer faults and/or reduced probability of faults causing failures, particularly if new modules are better developed and documented than the existing ones. Redeveloped software may meet requirements more closely and/or be structured to minimize errors. Modules can be redesigned so that high exposure functions are isolated in smaller, less complex and thoroughly tested modules with reduced failure likelihood. Re-engineered code can also reduce future maintainability risk. However, it is

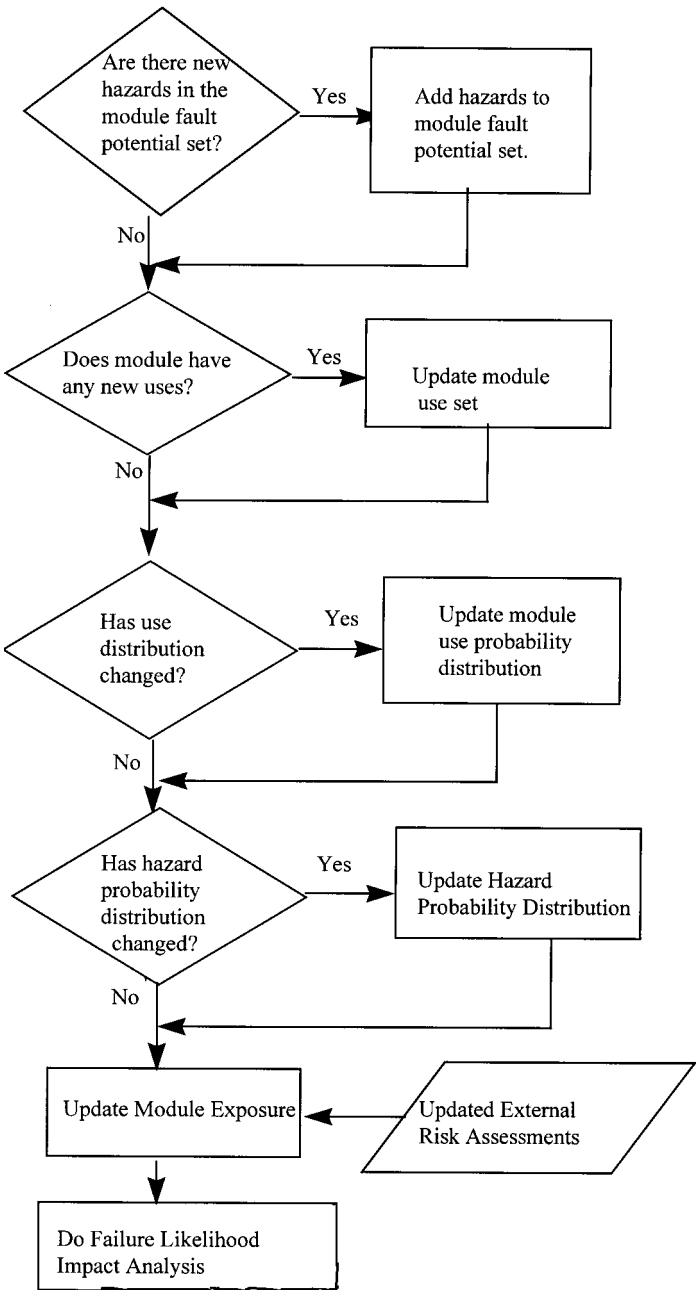


Figure 2. Procedure for module exposure impact analysis for revised modules

important to note that exposure can increase if new external procedures are required to co-ordinate output from the old and new modules.

4.5. An example of risk impact analysis

We illustrate the procedures for analysing the impact of maintenance on risk with a commercial loan system used by a savings and loan association. Perfective maintenance is required to add retail loan processing functionality to the existing system. We demonstrate how this will affect external failure and then we illustrate the impact of these changes on the exposure of one of the major modules in the system, a very large module that is used for all financial posting. First, updated failure likelihood and risk assessments are estimated for the addition of the required new code to this module. Second, we analyse an alternative maintenance proposal that would redesign the complete financial posting module. We illustrate the impact of redevelopment on failure likelihood and module risk.

Failure risk for the commercial loan system prior to maintenance was evaluated in (Sherer, 1992a, pp. 87–114). The processing of new loans introduces new external hazards that are similar to commercial loans, but the control risks and consequences differ. Retail loans tend to be smaller in value, on average, than commercial loans. Also, it is expected that the control risk associated with users not recognizing billing errors is higher for retail loans compared with commercial loans. Table 2 provides some comparisons for the magnitude of processing and control risk in retail versus commercial loans. The estimated consequences of the new external hazards, those associated with retail loans, are shown in Table 3, compared with consequences of hazards associated with commercial loans.

Our first maintenance proposal adds all retail loan financial payment processing to the current financial posting module in the commercial loan system. We relate this module to the new hazard consequences associated with these loans. The new use distribution, shown in Table 4, is based upon two assumptions: 1—that one half as many retail loans will be processed compared with commercial loans, and 2—that the total proportion of use of the financial posting module is approximately 0·77 (which means that the module

Table 2. Comparison of retail and commercial loan processing

	Commercial	Retail
Number of loans	1 700	900
Average value of loan	$170 000	$17 000
Average monthly invoices	$26 700 000	$1 400 000
Control risks:		
Probability loan output control does not recognize missing invoices	0·1	0·1
Probability customer does not recognize/report missing invoices	0·1	0·25
Consequence of hazard:		
Not producing customer invoices	$267 000/month	$35 000/month

Table 3. Hazard consequences

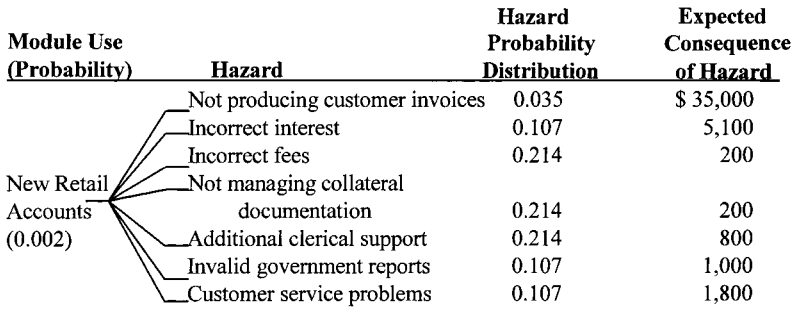
Hazard	Hazard consequence ($/month)	
	Commercial loans	Retail loans
Not producing customer invoices	267 000	35 000
Invalid interest	29 000	5 100
Invalid fees	3 000	200
Incorrect tracking of payments, disbursements	5 300	2 100
Invalid access	700	500
Insufficient collateral	850	400
Misposting to general ledger	1 000	1 000
Customer service problems	3 000	1 800
Not managing collateral documentation	600	200
Additional clerical support	800	800
Invalid government reports	1 000	1 000

Table 4. Exposure assessment: adding retail loan processing to financial posting module

Module use	Probability of use	Expected consequence of all hazards ($/month)
Commercial loan payments	0·482	1 999
Commercial loan advances, takedowns	0·010	10 738
New commercial loan accounts	0·005	13 992
Collateral set-up for commercial loans	0·005	2 608
Other data entry for commercial loans	0·010	9 783
Retail loan payments	0·241	1 260
Retail loan advances, takedowns	0·005	2 014
New retail accounts	0·002	2 327
Collateral set-up for retail loans	0·002	705
Other data entry for retail loans	0·005	1 940

Module exposure = Σ(probability of use)(expected consequence of all hazards).
Module exposure = $1 581/month.

is used only about three-quarters of the time that the system is used and that the remaining use of the system is primarily for data inquiries that do not involve financial posting). Expected consequences of all hazards associated with each use of the module, shown in Table 4, were estimated as the product of the consequence of each hazard in the module fault potential set and its hazard probability. An example of these computations is illustrated for new retail accounts in Figure 3. Our assumption regarding the hazard probability distribution is that, should a failure occur, minor hazards with relatively small consequences would be six times as likely as major hazards and twice as likely as hazards with average consequences, because software developers and maintainers are implicitly aware of the failure consequences, and subsequently, subject code with higher failure consequence to more stringent testing (Sherer, 1992a, p. 103).



$$\text{Expected Consequence of all hazards} = \sum(\text{Hazard Probability})(\text{Expected Consequence of Hazard}) = \$2327$$

$$\text{Exposure from Use} = (\text{Probability of Use})(\text{Expected Consequence of all Hazards}) = \$4.7$$

Figure 3. An event tree representation of the hazard consequences computation for the new retail accounts example

When we evaluate module exposure with the new use distribution and hazard consequences for all uses of this module, the expected exposure, given failure of this module, is $1 581/month compared with $1 879/month before maintenance, as shown in Table 5. Exposure is reduced because, one third of the time, the savings and loan will be processing smaller loans. If a failure occurs when processing these loans, the consequence will not be as great as failure associated with processing commercial loans.

While exposure decreases, the failure likelihood, or expected number of failures within one month, increases with changes to the financial posting module. Estimates are shown in Table 5 for Proposal 1. The mean number of faults increases because there are expected to be more faults in the newly added code. The per fault failure rate decreases somewhat because the additional code increases the number of possible paths in the code, reducing the opportunity for a particular fault to be encountered. The execution time per month also increases because more loans will be processed. The expected number of failures is computed with the new parameters and time translated to account for the number of failures experienced in the original code. It includes failures expected from the original code after use for eight months plus the expected number of failures from faults in the maintained code during the first month of its use. Expected number of failures therefore increases. Failure risk increases from $1 404/month to $1 928/month with the addition of the new code, an increase of $524/month.

Proposal 2 involves substantial redesign and restructuring of the code into 29 independent modules, control modules as well as modules performing independent functions, e.g., posting one of four types of collateral for commercial loans or setting up new retail accounts. By structuring cohesive modules with minimal coupling, we can isolate the high exposure functions. For example, a single module that will process new commercial loan accounts has an exposure of $70/month. Since this module is much smaller than the complete financial posting module in the original system, it is expected to have fewer

Table 5. Software failure risk impact: financial posting module

	Number of modules	Module exposure $/month	Expected number faults per module	Per fault failure rate	Execution time per month per module	Expected number failures per module	Failure risk $/month
Before maintenance	1	1 879	23	0·0015	31·4	0·747	**1 404**
After maintenance							
Proposal 1	1	1 581	31	0·0011	46·5	1·22	**1 928**
Proposal 2	29						**215**
All loans	1	1 581	1	0·0050	0·5	0·002	3·2
Commercial	1	1 251	1	0·0050	0·5	0·002	2·5
Payments	3	963	3	0·0026	8·0	0·062	59·7
Advances	3	107	3	0·0026	0·3	0·002	0·2
New Accounts	1	70	3	0·0026	0·3	0·002	0·1
Collateral	5	13	3	0·0026	0·1	0·001	0·01
Other	1	98	3	0·0026	1·0	0·008	0·8
Retail	1	330	1	0·0050	0·3	0·001	0·3
Payments	3	304	3	0·0026	4·0	0·031	9·4
Advances	3	10	3	0·0026	0·2	0·002	0·02
New Accounts	1	5	3	0·0026	0·2	0·002	0·01
Collateral	5	1	3	0·0026	0·1	0·001	0·001
Other	1	10	3	0·0026	0·5	0·004	0·04

faults, although a higher per fault failure rate. The estimates for the number of faults and per fault failure rates were based upon modules of similar size developed by the same developer (Sherer, 1992a, pp. 105–106). Table 5 shows the module risk estimates for the new modules that would replace the existing financial posting module.

Overall risk in the restructured system for Proposal 2 is $215/month compared with $1 404/month for the current system, and $1 928/month for Proposal 1 which added the new functionality within the existing code. Because Proposal 2 results in a system that is made up of small cohesive modules that are easier to debug and test, it is expected to have fewer faults, thereby reducing risk. The analysis shows the contribution of redesign and restructuring software to decrease software failure risk.

5. MAINTENANCE MANAGEMENT

5.1. Functions of maintenance management

Now that we have demonstrated that we can measure the impact of proposed maintenance on software failure risk, we extend the analysis to all the maintenance risks. First we will discuss the functions of maintenance management. Then we will indicate how maintenance management decisions can be guided by the impact of maintenance on all maintenance risks, including software failure risk.

Management is the process of achieving organizational goals through:

- Planning—setting goals and deciding how best to achieve them;
- Organizing—allocating and arranging resources to carry out plans;
- Leading—influencing others to engage in behaviours needed to reach organizational goals; and
- Controlling—regulating activities so that actual performance conforms to expected organizational standards and goals.

Planning, organizing and controlling maintenance activities can be aided by explicit analysis of the various risks in maintenance, including software failure risk. Understanding the maintenance risk profile requires managers to first assess the different risks and then manage those risks. Maintenance managers should make decisions that take into account the risk profile. For example, high project risk projects might require tighter project management controls, such as additional milestone meetings, whereas high usability and maintainability risk projects might suggest redevelopment of the system.

Maintenance management decisions involve trade-offs of the various maintenance risks. However, while many maintenance projects carry out some form of change impact analysis in order to scope the size of a change and consider alternative solutions, there has been 'a general lack of an explicit risk management process for maintenance projects' (Hinley and Bennett, 1993). Maintenance resource problems have often been associated with a lack of priority-based plans; few managers provide a rational basis for resource scheduling (Hinley and Bennett, 1993). Practical decisions are often based on gut feelings, what feels right for people involved (Tsivkin, 1996).

External risk impact analysis can help to determine whether to proceed with maintenance or redevelop the system. Throughout the maintenance process, the risk impact analysis can guide the allocation of personnel, time and money, and help plan release schedules.

5.2. Planning maintenance

5.2.1. Benefits and costs

This section develops and illustrates a model that managers can use to determine the benefits with proposed maintenance projects in terms of the impact of the maintenance on risk. Managers can then compare these benefits to costs to determine whether to proceed with particular maintenance projects. The proposal to redevelop the commercial loan system while incorporating retail loan processing is used to illustrate the model.

Maintenance should generally be accomplished only if benefits exceed costs. Generally, we redevelop a system instead of maintaining it only when the pain of making the next change becomes acute (Pfleeger and Bohner, 1990). Complexity metrics are sometimes used to indicate whether a program has degraded to such a point (Stark and Oman, 1995; Welker, Oman and Atkinson, 1997). Sakthivel (1994) has developed a model to compare costs and benefits to determine whether to redevelop or maintain a system. While this model includes various opportunity costs, it does not explicitly include software failure risk.

The benefits of maintenance come from decreased overall usability and maintainability risk, including the decreased functionality risk that may result from adding new functionality. Maintainability risk can increase with maintenance because of deterioration cost

(Sakthivel, 1994). Maintenance efforts often increase the complexity of the system, making it more difficult to maintain in the future. Increases in maintainability risk can be estimated from change management indicators, such as impact/scope, size, complexity, testability and traceability (Bohner, 1996). Functionality, performance and financial risk often decrease via maintenance efforts, especially perfective maintenance, where the addition of new features and applications add benefits to the user. However, these risks could increase, especially with corrective maintenance because of obsolescence resulting from not implementing a more usable substitute—i.e., one with additional functionality or improved performance (Sakthivel, 1994). Thus, the changes in each of these risks are the increased benefits from perfecting the software minus obsolescence costs. Software failure risk may either increase or decrease after maintenance as determined by the failure risk impact assessment. The cost of maintenance is the project cost adjusted for project risk. The latter is an estimate of the expected additional cost if the project is not completed satisfactorily.

5.2.2. *Cost/benefit model*

We now provide a model that can be used to determine whether to proceed with the maintenance project. Our objective is for an overall reduction in risk, where benefits exceed the cost, as expressed:

$$|\Delta FR + \Delta PR + \Delta \$R + \Delta SFR + \Delta MR| > EPC \tag{2}$$

and

$$\Delta FR + \Delta PR + \Delta \$R + \Delta SFR + \Delta MR < 0 \tag{3}$$

where ΔFR = change in functionality risk ($FR_{\text{after}} - FR_{\text{before}}$), ΔPR = change in performance risk, $\Delta \$R$ = change in financial risk, ΔSFR = change in software failure risk, ΔMR = change in maintainability risk and EPC = expected project cost adjusted for project risk.

5.2.3. *Model application*

We illustrate the model with an extension of our analysis to add retail loan processing to the commercial loan system while redeveloping the excessively large and complex processing modules. There are four such modules in the current commercial loan system. These include the financial posting module introduced in Section 4, as well as modules for billing extraction, billing print and report extraction. We perform a cost/benefit analysis for maintenance that will restructure these four modules into new, independent, cohesive and much smaller modules, incorporating retail loan functioning.

Table 6 summarizes the risk impact of these changes. Our assumptions in assessing these risks are discussed below.

Functionality risk. This is reduced by the addition of new retail loans. It is assumed to decrease by $5 000/month, which is an estimate of the additional income to be earned from processing the new retail loans.

Performance and financial risk. We assume that maintenance will not impact performance or financial risk.

Table 6. Impact of maintenance on risk and cost: commercial loan system

	ΔRisk ($/month)
Functionality risk	5 000
Performance risk	0
Financial risk	0
Software failure risk	−3 029
Maintainability risk	−752
Total impact on risk	−$8 781 per month
Expected project cost	$4 888 per month

Software failure risk. In Section 4, we showed that the impact on software failure risk for restructuring one of the large batch processing modules, the financial posting module, was $1 189/month ($1 404 − $215). We assume that similar restructuring will occur in the other three large batch processing modules, reducing risk in these modules by $1 118/month, $361/month and $361/month. Several new on-line processing modules will be required for the new retail loans. However, these modules are expected to have minimal failure risk since they are small modules with low failure likelihood and little exposure. Total reduction in software failure risk is therefore $3 029/month.

Maintainability risk. For the financial posting module, the expected number of failures per month before maintenance was estimated to be 0·747. Assuming that the cost of fixing a fault related to a failure is $400, maintainability risk for this module is estimated at approximately $300/month prior to maintenance. After maintenance, the expected number of failures for all the new modules shown in Table 5 total 0·32. Since each module is much smaller and less complex, it is assumed that the cost to fix each fault is only $350/month, resulting in an expected maintenance cost of $112, reducing maintainability risk by $188/month. Assuming similar reductions in the other three large batch processing modules that will be redeveloped, total maintainability risk reduction is $752/month.

Project risk. Project risk can be estimated as a percentage of cost. For example, let us assume that the total project cost to add the retail loan processing functionality and restructure the four large modules is estimated to be $160 000. Assume there is a 10% chance that this project will not be completed on time or within budget. Expected total cost is then estimated as $176 000. Ignoring the time value of money and assuming a three year life, the expected project cost per month is $4 888.

Cost/benefit discussion. Since the cost is $4 888/month while the benefit is $8 781 per month, this appears to be a worthwhile investment. It is interesting to note that analysis of the impact of maintenance solely on functionality would have provided minimal management guidance, since reduced functionality risk is approximately equal to the cost. However, once the reductions in software failure and maintainability risk are included, there is much stronger support for proceeding with the project.

It is also important to note that risk is not always reduced via maintenance. Maintenance Proposal 1, introduced in Section 4, added some retail loan processing capabilities within

the existing code structure, which would have increased both software failure risk and maintainability risk as a result of the increase in the expected number of failures.

5.3. Organizing and controlling maintenance activities

Software failure risk assessment can not only help determine whether to perform maintenance or redevelop software, it can also guide the effort. Table 7 summarizes how the different steps in the risk impact methodology can support the maintenance tasks (Bohner, 1996).

Risk assessment can help with key decisions regarding allocation of resources. Papapanagiotakis and Breuer (1994) present a model to allocate personnel and software/hardware environments utilities to maintenance efforts. This model can assign personnel based upon complexity indices for the application software, allowing for a threshold for skills and abilities of the servers assigned to different tasks. We suggest consideration of the level of software failure risk when allocating resources so that high failure risk modules requiring change are assigned to more experienced maintainers.

Evaluation of the economic impact of maintenance has another advantage. The failure risk assessment can serve as a motivational factor. Maintenance has typically not been considered a valuable career opportunity because programmers feel that they are not advancing with new technology (Parikh, 1986, p. 137). Maintenance is frequently considered to be boring and thankless 'dogwork' (Vowler, 1992). When compared with new application development, improvements are less visible because the impact of code revision is not easily measured. Information regarding the economic significance of the failures whose likelihood is reduced during maintenance can provide motivation for maintenance.

Risk impact can also guide prioritization of maintenance activities. Given the large backlogs of both development and maintenance activities in many organizations, managers must effectively prioritize efforts. Changes that most improve the profitability of the business should be given highest priority. In practice, subjective ranking methods are the norm (Turver and Munro, 1994). For preventative maintenance especially, prioritization is often based upon 'worst–first maintenance' which focuses on the worst code, or the code upon which most maintenance effort has historically been expended (Weinberg, 1980). This assumes that past maintenance efforts are indicative of future maintenance

Table 7. Risk impact methodology supports software maintenance tasks

Maintenance task (Bohner, 1996)	Failure risk assessment
Identify software change impacts	Hazard identification
Examine requirements traceability	Hazard scenario analysis
Classify change and explore similar changes	Historical risk impact analysis
Determine requirements impacts	Consequence analysis
Software design impacts	Module fault potential
Source program impacts	Module exposure assessment
Design program changes	Module risk assessment impact
(Re)test affected software	Risk guided testing (Sherer, 1996)

requirements, an assumption that may not be valid. As functionality changes and software is fixed, prior poor performance may no longer predict future problems. Risk may be higher in a section of the code that has not previously been maintained.

The ability to estimate module failure risk provides the capability to perform 'high–risk first' preventative maintenance. When resources are limited, those portions of the software that exhibit the highest failure risk should be primary maintenance candidates. Modules may exhibit high failure risk because they have high failure likelihood; often corresponding with the 'worst' modules, those that historically required the highest maintenance efforts. Modules with low failure likelihood may exhibit extremely high exposure, warranting expenditure of maintenance efforts even though they may have failed infrequently. On the other hand, some modules that traditionally had high maintenance costs may have little exposure. Focusing maintenance efforts on these modules may keep personnel from performing high benefit development or maintenance activities with more significant potential pay-off.

Risk impact analysis can also help to determine optimal release schedules (Sherer, 1992b). Scheduled maintenance defers changes until a predetermined time when a group of corrections are installed together rather than immediately repairing as each maintenance request is received (Lindhorst, 1980). Scheduling and batching releases based upon change impact or ripple effect information can reduce both project costs and defects (Turver and Munro, 1994). Batching corrections reduces resources needed to prepare, disseminate and install corrections. Moreover, scheduling maintenance has several benefits not easily quantifiable, which include: consolidation of requests, programmer job enrichment, forcing user departments to think more about request changes, periodic application valuation, elimination of the 'squeaky wheel syndrome', programmer backup, better planning and recognition that data processing change requests are as important as user requests (Lindhorst, 1980). Finally, design errors may be more obvious when the corrections are implemented together.

For corrective maintenance, managers often must consider whether to perform the maintenance on an emergency basis or schedule it for a future release. Users who wait to install corrections are frequently aware of the cost and risk of immediate fixes, but are not persuaded of their benefits. Software failure risk provides an assessment of the risk that is averted. The benefit from fixing errors immediately is the decreased failure risk during the time period until maintenance is scheduled. If the additional cost to fix a module immediately is less than the operational failure risk, then the maintenance manager should consider this course of action. Of course, the additional non-quantifiable benefits from scheduled maintenance should also be considered. On the other hand, scheduled maintenance is generally desirable if the operational failure risk is less than the additional cost of immediately implementing the changes.

6. CONCLUSIONS

Software maintenance management efforts can be improved through explicit consideration of maintenance risks. These risks include project risk, maintainability risk and usability risks, such as functionality, performance, financial and software failure risk. In particular, the measurement of the effect of maintenance on software failure risk can be used to

guide maintenance decisions. Alternatives to maintenance, such as redesign or redevelopment should be considered if the reduction in risk does not exceed the cost of maintenance. An understanding of the effect of maintenance can also guide decisions regarding the allocation of resources and the prioritization of maintenance efforts.

References

Arnold, R. and Bohner, S. (1993) 'Impact analysis—towards a framework for comparison', in *Proceedings of the Conference on Software Maintenance*, Montreal, Quebec, Canada, IEEE Computer Society Press, Los Alamitos CA, pp. 292–301.

Ash, D., Alderete, J., Yao, L., Oman, P. and Lowther, B. (1994) 'Using software maintainability models to track code health', in *Proceedings of the International Conference on Software Maintenance*, Victoria, British Columbia, Canada, IEEE Computer Society Press, Los Alamitos CA, pp. 154–160.

Bassett, P. (1995) 'Software springs to life: software maintenance', *Computing Canada*, **21**(13), 42.

Bohner, S. (1996) 'Impact analysis in the software change process: a year 2000 perspective', in *Proceedings of the International Conference on Software Maintenance*, Monterey, Canada, IEEE Computer Society Press, Los Alamitos CA, pp. 42–51.

Bohner, S. and Arnold, R. (1996) *Software Change Impact Analysis*, IEEE Computer Society Press, Los Alamitos, CA, 376 pp.

Brooks, F. P. (1987) 'No silver bullet: essence and accidents of software engineering', *Computer*, **20**(14), 10–19.

Deklava, S. (1992) 'Delphi study of software maintenance problems', in *Proceedings of the Conference on Software Maintenance*, Orlando FL, IEEE Computer Society Press, Los Alamitos CA, pp. 10–17.

Drew, D. (1992) 'Tailoring the Software Engineering Institute's (SEI) Capability Maturity Model (CMM) to a software sustaining engineering organization', in *Proceedings of the Conference on Software Maintenance*, Orlando FL, IEEE Computer Society Press, Los Alamitos CA, pp. 137–144.

Henry, J., Blasewitz, R. and Kettinger, D. (1996) 'Defining and implementing a measurement–based software maintenance process', *Journal of Software Maintenance*, **8**(2), 79–100.

Hinley, D. and Bennett, K. (1992) 'Developing a model to manage the software maintenance process', in *Proceedings of the Conference on Software Maintenance*, Orlando FL, IEEE Computer Society Press, Los Alamitos CA, pp. 174–182.

Hinley, D. and Bennett, K. (1993) 'Reducing the risks in software improvement through process–orientated management', in *Proceedings of the International Conference on Software Maintenance*, Montreal, Quebec, Canada, IEEE Computer Society Press, Los Alamitos CA, pp. 319–328.

Lanubile, F. and Visaggio, G. (1995) 'Decision–driven maintenance', *Journal of Software Maintenance*, **7**(2), 91–115.

Leung, H. (1996) 'A risk index for software producers', *Journal of Software Maintenance*, **8**(5), 281–294.

Lindhorst, W. (1980) 'Scheduled program maintenance', in Parikh, G. (Ed), *Techniques of Program and System Maintenance*, Ethnotech, Inc., Lincoln NE, pp. 133–136.

Moad, J. (1990) 'Maintaining the competitive edge', *Datamation*, **36**(4), 61–62, 66.

Musa, J., Iannino, A. and Okumoto, K. (1987) *Software Reliability: Measurement, Prediction, Application*, McGraw–Hill Book Co., New York NY, 621 pp.

Papapanagiotakis, G. and Breuer, P. (1994) 'A software maintenance management model based on queuing networks', *Journal of Software Maintenance*, **6**(2), 73–97.

Parikh, G. (1986) *Handbook of Software Maintenance*, John Wiley & Sons, Inc., New York NY, 421 pp.

Pearse, T. and Oman, P. (1995) 'Maintainability measurements on industrial source code maintenance activities', in *Proceedings of the International Conference on Software Maintenance*, Opio (Nice), France, IEEE Computer Society Press, Los Alamitos CA, pp. 295–303.

Peercy, D. (1981) 'A software maintainability evaluation methodology', *IEEE Transactions on Software Engineering*, **7**(4), 343–351.
Pfleeger, S. (1991) *Software Engineering*, Macmillan Publishing Company, New York NY, 517 pp.
Pfleeger, S. and Bohner, S. (1990) 'A framework for software maintenance metrics', in *Proceedings of the Conference on Software Maintenance—1990*, San Diego CA, IEEE Computer Society Press, Los Alamitos CA, pp. 320–327.
Pigoski, T. and Looney, C. (1993) 'Software maintenance training: transition experiences', in *Proceedings of the Conference on Software Maintenance*, IEEE Computer Society Press, Los Alamitos CA, pp. 314–318.
Sakthivel, S. (1994) 'A decision model to choose between software maintenance and software redevelopment', *Journal of Software Maintenance*, **6**(3), 121–143.
Sherer, S. A. (1992a) *Software Failure Risk: Measurement and Management*, Plenum Press, New York NY, 276 pp.
Sherer, S. A. (1992b) 'Cost benefit analysis and the art of software maintenance', in *Proceedings of the Conference on Software Maintenance*, Orlando FL, IEEE Computer Society Press, Los Alamitos CA, pp. 70–77.
Sherer, S. A. (1996) 'Statistical software testing using economic exposure assessments', *Software Engineering Journal*, **11**(5), 293–298.
Sneed, H. (1995) 'Estimating the costs of software maintenance tasks', in *Proceedings of the International Conference on Software Maintenance*, Opio (Nice), France, IEEE Computer Society Press, Los Alamitos CA, pp. 168–181.
Stark, G. (1996) 'Measurements for managing software maintenance', in *Proceedings of the International Conference on Software Maintenance*, Monterey CA, IEEE Computer Society Press, Los Alamitos CA, pp. 152–161.
Stark, G. and Oman, P. W. (1995) 'A survey instrument for understanding the complexity of software maintenance', *Journal of Software Maintenance*, **7**(6), 421–444.
Swanson, E. B. (1976) 'The dimensions of maintenance' in *Proceedings of the 2nd International Conference on Software Engineering*, San Francisco CA, IEEE Computer Society Press, Los Alamitos CA, pp. 492–497.
Swanson, E. B. and Beath, C. M. (1990) 'Departmentalization in software development and maintenance', *Communications of the ACM*, **33**(6), 658–667.
Taylor, M. and Wood–Harper, A. (1996) 'Methodologies and software maintenance', *Journal of Software Maintenance*, **8**(5), 295–308.
Tsivkin, V. (1996) 'Live short and prosper: applying cost analysis to routine software maintenance procedures', *Journal of Software Maintenance*, **8**(4), 257–267.
Turver, R. and Munro, M. (1994) 'An early impact analysis technique for software maintenance', *Journal of Software Maintenance*, **6**(1), 35–52.
Vogel, P. (1996) 'Death by development, application development cost management', *Datamation*, **42**(6), 108.
Vowler, T. (1992) 'Cinderella goes to the ball: the importance of software maintenance', *Computer Weekly*, 19 March, 1992, 26.
Weinberg, G. M. (1980) 'Worst first maintenance', in Parikh, G. (Ed), *Techniques of Program and System Maintenance*, Ethnotech, Inc., Lincoln NE, pp. 119–121.
Welker, K. D., Oman, P. W. and Atkinson, G. G. (1997) 'Development and application of an automated source code maintainability index', *Journal of Software Maintenance*, **9**(3), 127–159.

Author's biography:

Susan A. Sherer is an Associate Professor of Business at Lehigh University where she serves as the Director of the Information Systems Curriculum and Vice–Chair of the Business Department. Sue has managed software development and maintenance projects in several companies. Her research interests include software failure risk measurement, information systems risk management and risk in manufacturing networks. She is the author of a book and numerous papers. Sue's Ph.D. is in Decision Sciences from the Wharton School of the University of Pennsylvania.